操作系统408大纲

一、操作系统概述

(一)操作系统的概念、特征、功能和提供的服务

(二)操作系统的发展与分类

(三)操作系统的运行环境

1.内核态与用户态

2.中断、异常

3.系统调用

(四)操作系统体系结构

二、进程管理

(一)进程与线程

1.进程概念

2.进程的状态与转换

3.进程控制

4.进程组织

5.进程通信

共享存储系统；消息传递系统；管道通信。

6.线程概念与多线程模型

(二)处理机调度

1.调度的基本概念

2.调度时机、切换与过程

3.调度的基本准则

4.调度方式

5.典型调度算法

先来先服务调度算法；短作业(短进程、短线程)优先调度算法；时间片轮转调度算法；优先级调度算法；高响应比优先调度算法；多级反馈队列调度算法。

(三)同步与互斥

1.进程同步的基本概念

2.实现临界区互斥的基本方法 软件实现方法；硬件实现方法。

3.信号量

4.管程

5.经典同步问题

生产者-消费者问题；读者-写者问题；哲学家进餐问题。

(四)死锁

1.死锁的概念

2.死锁处理策略

3.死锁预防

4.死锁避免

系统安全状态，银行家算法。

5.死锁检测和解除

三、内存管理

(一)内存管理基础

1.内存管理概念

程序装入与链接；逻辑地址与物理地址空间；内存保护。

2.连续分配管理方式

3.非连续分配管理方式

分页管理方式；分段管理方式；段页式管理方式。

(二)虚拟内存管理

1.虚拟内存基本概念

2.请求分页管理方式

3.页面置换算法

最佳置换算法(OPT)；

先进先出置换算法（FIFO）；

最近最少使用置换算法(LRU)；

时钟置换算法(CLOCK)。

4.页面分配策略

5.工作集

6.抖动

四、文件管理

(一)文件系统基础

1.文件概念

2.文件的逻辑结构

顺序文件；索引文件；索引顺序文件。

3.目录结构

文件控制块和索引节点；单级目录结构和两级目录结构；树形目录结构；图形目录结构。

4.文件共享

5.文件保护

访问类型；访问控制。

(二)文件系统实现

1.文件系统层次结构

2.目录实现

3.文件实现

(三)磁盘组织与管理

1.磁盘的结构

2.磁盘调度算法

3.磁盘的管理

五、输入输出(I/O)管理

(一)I/O 管理概述

1. I/O 控制方式
2. I/O 软件层次结构

(二)I/O 核心子系统

1. I/O 调度概念
2. 高缓存与缓冲区
3. 设备分配与回收
4. 假脱机技术(SPOOLing)

一、操作系统引论

OS的定义与作用

定义

操作系统（Operating System，OS）是配置在计算机硬件上的第一层软件，是对硬件系统的首次扩充。

作用

- 作为用户与计算机硬件系统之间的接口
- 作为计算机系统资源的管理者
- 实现了对计算机资源的抽象

OS的发展过程

第一代计算机 :真空管和插件板(无操作系统)

人工操作方式

- 程序员将事先已穿孔的纸带（**对应程序和数据**），将穿孔的纸带装入纸带输入机，再启动输入机将纸带上的程序和数据输入计算机，然后启动计算机运行。仅当程序运行完毕并取走计算结果后，才允许下一个用户上机

缺点:

1. 用户独占全机，计算机的全部资源都由上机用户所独占
2. CPU会等待人工操作，在装卡、卸卡时，CPU及内存是空闲的

脱机输入/输出(Off-Line I/O)方式

- 在外围机的控制下，把纸带上的数据输入到磁带上。当CPU需要这些程序和数据时，再从磁带上高速地调入内存
- 输出同样如此，先由CPU把数据直接从内存高速地输送到磁带上，然后在一台外围机的控制下，将磁带上的结果通过相应的输出设备输出。

优点:

1. 通过这种脱机I/O方式，减少了CPU的空闲时间（人工操作的装卡、卸卡都是在外围机完成的，并不占用主机的时间）
2. 提高了I/O的速度（当需要输入数据时候，直接从高速的磁带上面输入到内存）

第二代计算机:晶体管和批处理系统

单道批处理系统

运行过程

- 在系统运行过程中，内存中只有一个用户作业存在
- 先把一批作业脱机输入到磁带上
- 系统配上监督程序
- 在监督程序控制下使这批作业能一个接一个的连续得到处理
- 处理机使用权在监督程序和用户程序间转换

特征

- 自动性（会通过使用权的互相转换，处理任务）
- 顺序性（作业只能一个接一个按照顺序得到处理）
- 单道性（内存中只能有一个作业被处理）

缺点

资源得不到充分的利用，因为在内存中只有一道程序，在发出I/O请求后，CPU便处于等待状态，必须在I/O完成后才可以继续运行。

第三代计算机：集成电路和多道程序设计（引入分时系统）

多道批处理系统

运行过程

- 作业先存放在外存上并排成一个队列，称为“后备队列”；然后，由作业调度程序按一定的算法从后备队列中选择若干个作业调入内存，**使它们共享CPU和系统中的各种资源。**可以在运行程序A时，因I/O操作而暂停执行时的CPU空档时间，再调度另一道程序B运行，以此类推，这样便可以保持CPU处于忙碌转态

特征

- 多道性（同一时间可以处理多个任务）
- 成批处理（会从后备队列中选择一批任务进行处理）
- 无序性（会按照一定的算法从后备队列中选择若干个任务进行处理）
- 调度性（在A任务处理中的空档时间，可以再调度B任务，以此类推）

优缺点

- 资源利用率高
- 系统吞吐量大
- 平均周转时间长
- 无交互能力
- 适合大型科学计算、数据处理。

多道批处理需要解决的问题

(1) 处理机管理问题。

(2) 内存管理问题。

(3) I/O设备管理问题。

(4) 文件管理问题。

(5) 作业管理问题。

(6) 用户与系统的接口问题

一些概念

吞吐量：单位时间（1h）内系统所处理的作业个数

周转时间：从作业进入系统到作业完成退出系统所用的时间

平均周转时间：同时参与系统运行的几个作业的周转时间的平均值。

分时系统

满足的需求

- 人机交互
- 共享主机
- 便于用户上机

分时系统示意图

运行过程

- 一台主机连接了若干个终端
- 每个终端有一个用户在使用
- 交互式的向系统提出命令请求
- 系统接受每个用户的命令
- 采用时间片轮转方式处理服务请求
- 并通过交互方式在终端上向用户显示结果
- 用户根据上步结果发出下道命令

特征

- 多路性（可以允许多个用户同时共享一台计算机）
- 独立性（每个用户都在各自的终端操作，互不影响，就像是在独自体验一台计算机）
- 及时性（用户的请求会在很短时间内获得响应，这个等待时间是人们能接受的时间）
- 交互性（用户可以通过终端与系统进行广泛的人机对话，可以请求文件编辑、数据处理、访问文件系统、数据库系统，请求提供打印服务等）

实时系统

定义

- 是指系统能及时响应外部事件的请求，在规定时间内完成对该事件的处理，并控制所以实时任务协调一致地运行
- 及时性要求高，系统可靠性高

分类

- 实时控制系统
- 实时信息处理系统

实时任务

分类

- 按任务执行时是否呈现周期性可分为：

  周期性实时任务

  非周期性实时任务

- 根据对截止时间的要求可分为：

  硬实时任务（必须满足任务对截止时间的要求，否则可能出现难以预测的后果）

  软实时任务（偶尔错过了任务的截止时间，对系统产生的影响也不会太大）

实时系统与分时系统特征的比较

（1）多路性（实时系统周期性的对多路信息进行观察，或对多个对象进行控制）

（2）独立性（对信息的采集和对对象的控制互不干涉）

（3）及时性（硬实时任务和软实时任务，一般为秒级到毫秒级）

（4）交互性（仅限于访问系统中特定的专用服务程序）

（5）可靠性（要求系统高度可靠，否则会造成不可预料的后果）

微机操作系统的发展

单用户单任务：（只允许一个用户上机，且只允许用户程序作为一个任务运行）

- CP/M 8位操作系统
- MS-DOS 16位操作系统

单用户多任务操作系统:（只允许一个用户上机，但运行用户把程序分为若干个任务，使他们并发执行）

- windows 系列

多用户多任务操作系统:（运行多个用户通过各自的终端上机，共享主机各种资源，也同样可以使程序分为若干任务）（**Linux中的普通用户和root用户的终端权限不一样**）

- unix linux

OS的特征

并发性

（两个或多个事件在同一**时间间隔**内发生）（并行性：两个或多个事件**同一时间**发生）

共享性

互斥共享方式：（在一段时间内，只允许一个进程访问该资源）

- 打印机、磁带机

同时访问方式：（允许一段时间内由多个进程同时对其进行访问）

磁盘、文件

虚拟性

时分复用技术：（利用某设备为用户服务的空余时间，去为其它用户服务）

- 虚拟处理机技术
- 虚拟设备技术：如共享打印机

空分复用技术：（利用存储器的空闲空闲分区域存放和运行其它的多道程序）

- 虚拟磁盘技术
- 虚拟存储器技术

异步性

异步指的是让CPU暂时搁置当前请求的响应,处理下一个请求,当通过轮询或其他方式得到回调通知后,开始运行。

多道程序环境下程序的执行，是以异步方式进行的；每个程序在何时执行，多个程序间的执行顺序以及完成每道程序所需的时间都是不确定和不可预知的。进程是以人们不可预知的速度向前推进，此即进程的异步性。

OS的功能

处理机管理

主要任务：

对处理机的分配和运行实施有效管理。对处理机管理，可归结为对进程的管理。

主要功能：

- 进程控制：创建进程、撤消进程以及控制进程的状态转换。
- 进程同步：对并发执行的进程进行协调。最基本的方式是使诸进程以互斥方式访问临界资源。
- 进程通信：相互合作的进程间所进行的信息交换。
- 进程调度：按一定算法从进程就绪队列中选出一进程，把处理机分配给它，为该进程设置运行现场，并使之投入运行。

存储器管理

主要任务：

为多道程序的并发运行提供良好环境；

便于用户使用存储器；

提高存储器的利用率；

为尽量多的用户提供足够大的存储空间。

主要功能：

- 内存分配：为每道程序分配内存。
- 内存保护：保证各道程序都能在自己的内存空间运行而互不干扰。
- 地址映射：把程序地址空间中的逻辑地址转换为内存空间对应的物理地址
- 内存扩充：借助于虚拟存贮技术，使系统能运行内存要求量远比物理内存大得多得作业，或让更多得作业并发执行。

设备管理

主要任务：

为用户程序分配I/O设备；

完成用户程序请求的I/O操作；

提高CPU和I/O设备的利用率;

改善人机界面。

主要功能：

- 缓冲管理:缓和CPU和I/O设备间速度不匹配的矛盾，和提高CPU与设备、设备与设备间操作的并行程度。
- 设备分配:系统根据用户所请求的设备类型和所采用的分配算法对设备进行分配，并将未获得所需设备的进程放进相应设备的等待队列。
- 设备处理:实现CPU和设备控制器之间的通信。

- 虚拟设备功能:通过某种技术使独占设备成为能被多个用户共享的设备。

文件管理

主要任务:

对用户文件和系统文件进行管理，以方便用户使用，并保证文件的安全性。

主要功能:

- 文件存储空间的管理
- 目录管理
- 文件读、写管理
- 文件保护
- 向用户提供接口

用户接口

联机用户接口:

- 一组命令。

脱机用户接口:

- 作业控制语言编写作业说明书。

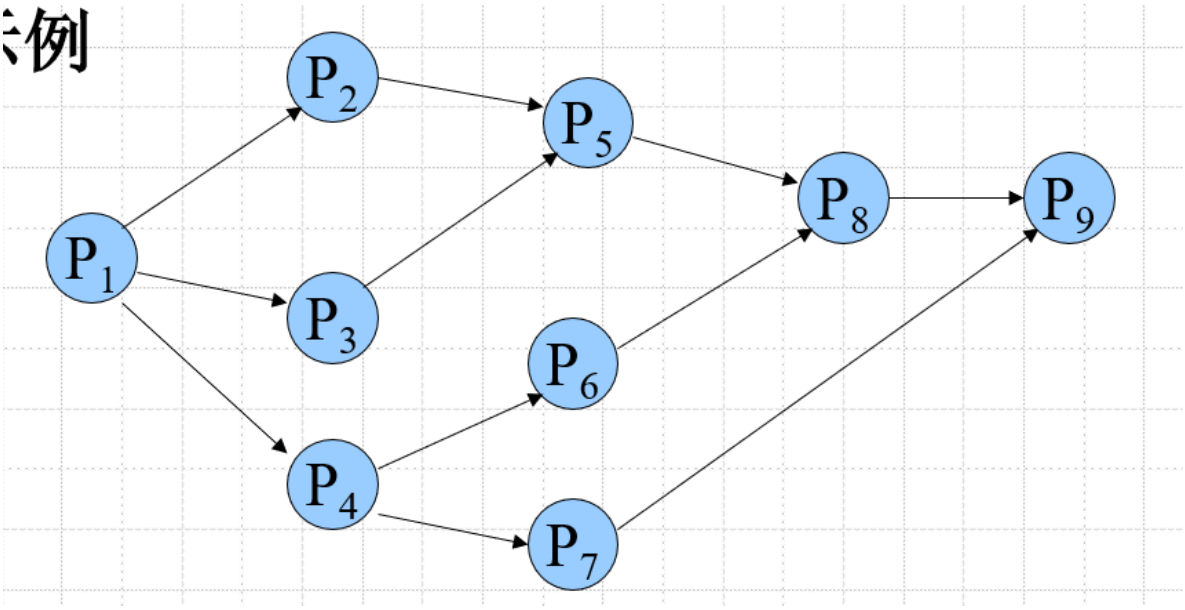
程序接口:

- 系统调用，API

二、进程的描述与控制

前趋图

前趋图是一个有向无循环图 pi->pj，表示进程pi和pj之间存在着前趋关系，即只有pi进程完成以后，pj进程才可以进行



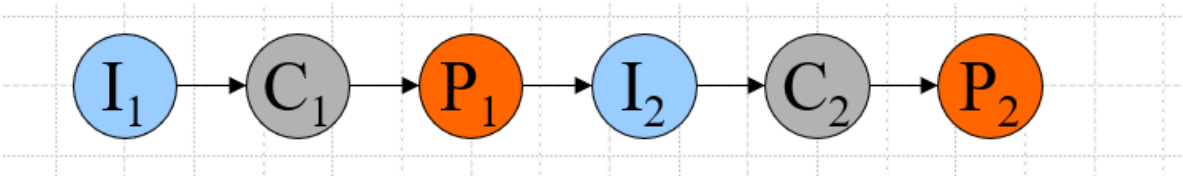
前趋关系：

P = {P1,P2,P3,P4,P5,P6,P7,P8,P9}

→={(P1,P2),(P1,P3),(P1,P4),(P2,P5),(P3,P5),(P4,P6),(P4,P7),(P5,P8),(P6,P8),(P7,P9),(P8,P9)}

程序的顺序执行

各程序必须分成若干个程序段，各程序段之间，必须按照某种先后次序顺序执行

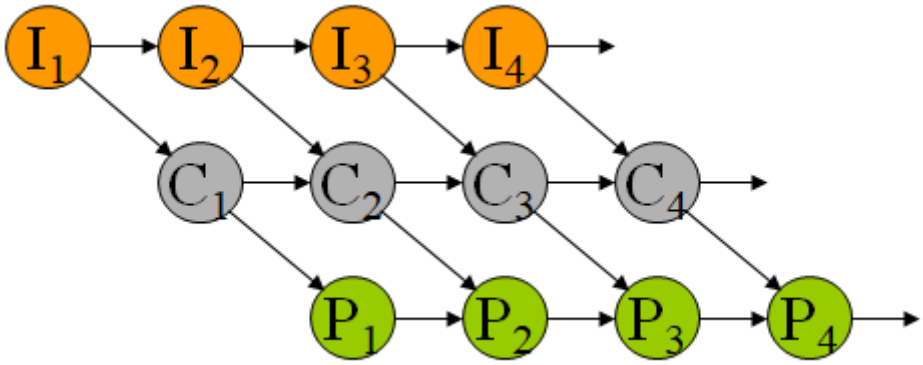


特征

- 顺序性（每一次操作必须在下一个操作开始之前结束）
- 封闭性（程序运行时独占全机资源）
- 可再现性（当程序重复执行时，不论是从头到尾还是怎么执行，最后都能获得相同的结果）

程序的并发执行

并发执行是指两个程序执行时间上是**重叠**的。凡是能由一组并发程序完成的任务，都能由相应的单个程序完成。



前趋关系：Ii→Ci，Ii→Ii+1，Ci→Pi，Ci→Ci+1，Pi→Pi+1

可以并发执行： Pi -1、Ci、 Ii+1

即当I1执行完毕以后，在进行C1操作的同时，就可以继续执行I2操作了

特征

- 间断性（程序之间相互制约（**共享资源、相互合作**）导致“执行——暂停——执行”）
- 失去封闭性（共享资源，某一程序在运行时，其环境都必然会受到其它程序的影响）
- 不可再现性（程序经过多次执行以后，虽然执行时的环境和初始条件相同，但得到的结果却各不相同）

进程的描述

进程是由程序段（代码段）、数据段、PCB三部分构成。

进程是具有独立功能的程序在一个数据集合上运行的过程，它是系统进行资源**分配**和**调度**的一个独立单位。

进程的特征

- 动态性（进程有一定的生命周期）
- 并发性
- 独立性（进程是一个能独立运行、独立获得资源和独立接受调度的基本单位）
- 异步性（进程按照各自独立的、不可预知的速度向前推进，不可预料它是怎么完成的）

进程与程序的关系

	进 程	程 序
概念	动态实体，强调执行过程	静态实体，是指令的有序集合
特征	并发性、独立性、异步性，是竞争计算机系统资源的基本单位	无并行特征，是静止的
两者联系	不同的进程可以共享同一个程序，只要对应的数据集不同	

进程的三种基本状态

就绪、执行、阻塞

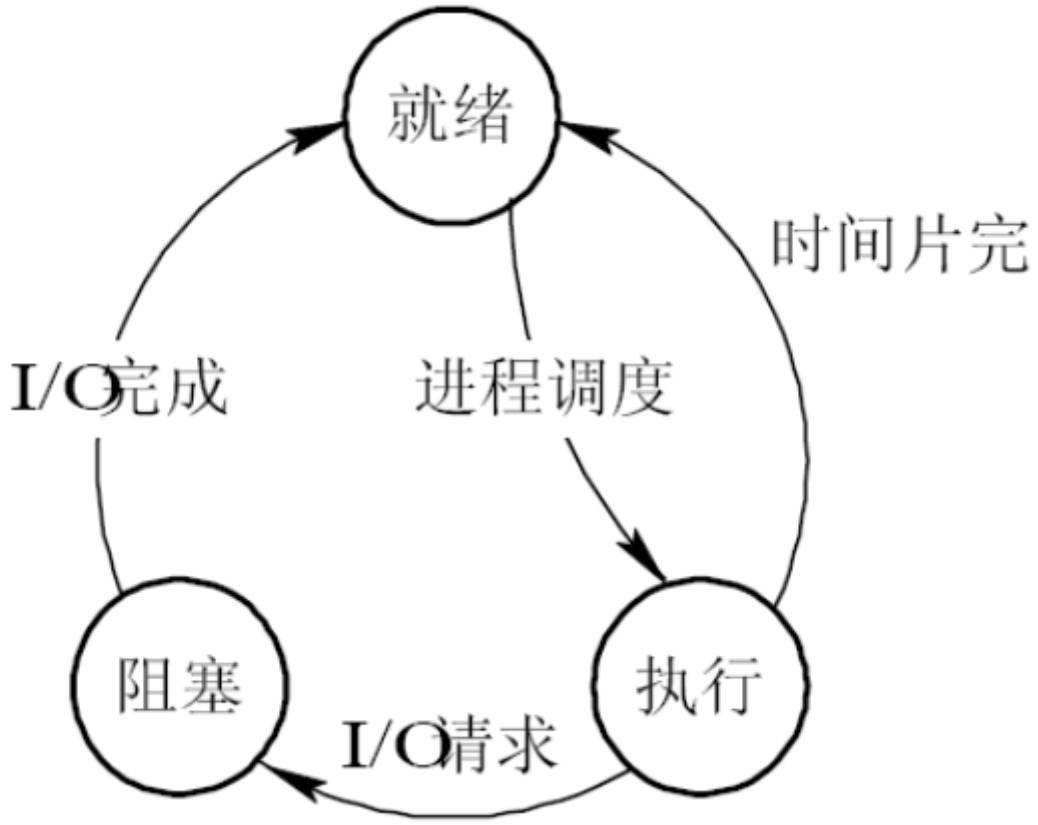
以在学校食堂打饭作为例子

就绪状态：代表准备好了饭盒、钱等一系列关于打饭的东西，就差轮到自己打饭了（除CPU以外的所有必要资源）

执行状态：正在打饭的过程（程序正在被执行）

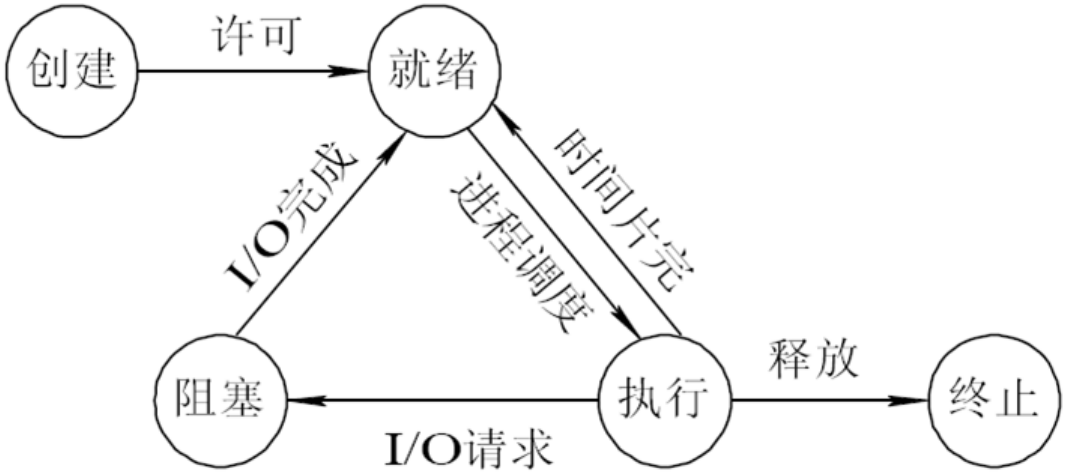
阻塞状态：轮到自己打饭的时候，发现自己的钱没带够，所以只能把钱拿够再去打饭，**而且也得重新排队**(因等待某事件发生而暂时无法继续执行，从而放弃处理机，使程序执行处于暂停状态)

进程基本状态转换图



即先进行就绪状态然后如果一切准备就绪那么将进入执行状态，如果发现自己缺少必要东西（钱不够）那么则进行阻塞状态，然后重新进入就绪状态（重新排队），重复以上操作。如果发现系统分配给它的时间片用完了而被剥夺处理机，那么要重新转换为就绪状态。

如果加上创建状态和终止状态以后进程基本状态转换图就变为



进程的五种基本状态及转换

临界区

人们把在每个进程中访问临界资源的那段代码称为临界区

同步机制

- 空闲让进
- 忙则等待
- 有限等待
- 让权等待

信号量机制

整型信号量

S仅能通过两个标准的原子操作(Atomic Operation) wait(S)和signal(S)来访问。这两个操作一直被分别称为P、V操作。 wait和signal操作可描述为：

```
wait（S）{
    while(S<=0);
    S--;
}
signal(S){
S++;
}
```

记录型信号量

```
struct semaphore{
        int value;
        struct process_control_block *L;
    }
wait(semaphore S)
{
      S.value--;
      if(S.value < 0)     block(S.L)
    }
signal(semaphore S)
```

```
{
    S.value++;
    if(S.value≤0)       wakeup(S.L);
}
```

经典进程的同步问题

1.生产者-消费者问题

问题描述

有一群生产者进程在生产产品，并将这些产品提供给消费者进程去消费。为使生产者进程与消费者进程能并发执行，在两者之间设置了一个具有n个缓冲区的缓冲池，生产者进程将它所生产的产品放入一个缓冲区中；消费者进程可从一个缓冲区中取走产品去消费。

算法描述

```
int in, out = 0;//产品对应进出的下标
item buffer[n];//缓冲区的大小为n
//mutex为互斥信号量，实现对缓冲区的互斥访问，初值为1，取值为0和1
//empty和full为资源信号量，为正时表示可用资源的数目；为负时，其绝对值为相应阻塞进程的数目。初始为n,0。可为负。
semaphore mutex = 1,empty = n, full = 0;
void producer(){
    do{
```

```
        producer an item nextp;//生产一个商品nextp
        ...
        wait(empty);//确认缓冲区是否未满
        wait(mutex);//对缓冲区进行互斥访问
        buffer[in] = nextp;//将生产的商品放入缓冲区
        in = (in+1) % n;//因为缓冲区为一个循环队列所以要模n
        //唤醒临界区以及告诉缓冲区增加一个产品，顺序可以颠倒
        signal(mutex);
        //唤醒消费者进程
        signal(full);
    }while(true);
}

void consumer(){
    do{
        wait(full);//判断缓冲区是否未空
        wait(mutex);//进行互斥访问
        nextc = buffer[out];//从缓冲区拿来一个商品进行消费
        out = (out+1) % n;
        signal(mutex);
        signal(empty);
        consumer the item in nextc;
        ...
    }while(true);
}
```

```
33 //wait两个操作不能颠倒顺序，因为如果交换顺序，就会先对缓冲区进行互斥访问，在判断缓冲区是否有空位，此时如果临界区已满，那么生产者进程堵塞，就会跳转到消费者进程，此时因为mutex还在生产者那里没有被释放，因此消费者也被堵塞，则会导致生产者在等待消费者对缓冲区的释放，而消费者在等待生产者释放临界区的情况。彼此循环等待，没有被唤醒，则会出现死锁现象。
34
35 //所以实现互斥的P操作，一定要在实现同步的P操作之后！！！
36 //signal操作即V操作，顺序无所谓，先释放临界区再释放缓冲区，与先释放缓冲区再释放临界区一样
37
38 wait (empty);//生产者判断缓冲区是否空闲（未满），消耗一个空闲缓冲区
39
40 wait (mutex);//消费者和生产者对临界区进行互斥访问
41
42 signal (full);//生产者唤醒消费者， 增加一个商品
43
44 signal (mutex);//释放缓冲区
45
46 wait (full);//消费者判断缓冲区是否不空闲（非空），消耗一个商品（非空缓冲区）
47
48 signal (empty);//消费者唤醒生产者， 增加一个空闲缓冲区
```

```
49
```

实现互斥是在同一进程中进行一对PV操作

实现两进程同步是在在其中一个进程执行P操作，另一个进程执行V操作

同步关系:

- 当缓冲区空时，消费者必须等待生产者生产商品
- 当缓冲区慢时，生产者必须等待消费者消费商品

2.读者-写者问题

一个文件可能被多个进程共享，为了保证读写的正确性和文件的一致性，系统要求，**当有读者进程读文件时，不允许任何写者进程写**，但允许多读者同时读；当有写者进程写时，不允许任何其它写者进程写，也不允许任何读者进行读。

互斥关系：

- 一个文件只能被一个写者写或者被多个读者进行读操作

同步关系:

- 当有读者进程的时候，写者进程必须进行等待
- 当有写者进程的时候，其他写者和读者必须进行等待

```
```

```
semaphore rmutex = 1, wmutex = 1;//创建两个互斥信号量，分别代表读和写
int readcount = 0;//表示正在读的进程数目
void reader(){
    do{
        wait(rmutex);//进行读操作互斥
        if (readcount == 0)wait(wmutex);//判断此时是否有写进程，如果有则进行等待操作，如果没有则阻止写进程，因为这是读者进程，不允许任何写者进程写。
        readcount++;
        signal(rmutex);
        ...
        perform read operation;
        ...
        //下面是最后一个的操作
        wait(rmutex);
        readcount--;//读完进程，读者数减一
        if(readcount == 0)signal(wmutex);//判断是否有写进程，如果有则唤醒写进程，也就是最后一个读者走了，就要唤醒写进程。
        signal(rmutex);//
    }while(true);
}

void Writer(){//只要没有读者，就可以进行写操作
    do{
        wait(wmutex);
```

```
23          perform write operation;
24          signal(wmutex);
25      }while(true);
26  }
```

3.老和尚与小和尚问题

某寺庙，有小和尚、老和尚若干．庙内有一水缸，由小和尚用桶提水入缸，供老和尚用桶从缸中取水饮用．水缸可容纳 30 桶水，每次入水、取水仅为1桶，不可同时进行。水取自同一井中，水井径窄，每次只能容纳一个水桶取水。设水桶个数为5个，试用信号量和 P、V 操作给出老和尚和小和尚的活动。

分析:

- 只有一个水缸
- 每次喝水取水都只能用一个水桶
- 水桶个数5个
- 总共有30桶水

互斥关系:

- 一个水井只能进行取水或者喝水操作
- 一个桶只能被用来取水或者喝水操作

同步关系:

- 当小和尚进行提水操作的时候，老和尚必须进行等待
- 当老和尚进行喝水操作的时候，小和尚必须进行等待

```
```

```
//定义变量
semaphore mutex_jar = 1,mutex_well= 1;//设定桶和水井的互斥变量
semaphore empty = 30, full = 0, buckets = 5;//根据题目要求赋初值


void younger(){//小和尚提水，需要判断有没有桶，且水井是不是被占用
    do{
        wait(empty);//判断水缸是不是空闲(未满)
        wait(bucket);//判断有没有桶
        go the well;
        wait(mutex_well);//互斥访问水井
        get water;
        signal(mutex_well);//释放临界区
        go to the temple;
        wait(mutex_jar);//对水缸进行互斥访问，因为喝水和取水不能同时进行
        pure the water into the jar;
        signal(mutex_jar);//释放水缸临界区
        signal(buckets);//释放水桶临界区
        signal(full);//水井里面的水加一，并通知老和尚进程
    }while(true);
}

void old_monk(){
    do{
```

```
25          wait(full);//判断水缸里面是否有水(未空)
26          wait(buckets);
27          wait(mutex_jar);
28          get water;
29          signal(mutex_jar);
30          signal(buckets);
31          signal(empty);//释放临界区资源，通知小和尚进程
32      }while(true);
33  }
```

三、处理机调度与死锁

处理机调度的层次(是以作业调度的周期长短命名)

	对象	用途
高级调度(长程调度)	作业	主要用于多批道处理系统中
中级调度(内存调度)		提高内存利用率和系统吞吐量
低级调度(短程调度)	进程	多批道处理、分时和实时都要配置

分时、实时系统不需要高级调度

低级调度是操作系统最核心的部分

作业调度

先来先服务(FCFS)算法

按照作业到达的先后顺序来进行调度

周转时间 = 完成时间 - 到达时间

短作业优先(SJF)算法

作业越短，其优先级越高

带权周转时间 = 周转时间 / 服务时间

进程调度

轮转调度算法

优先级调度算法

多队列调度算法

多级反馈队列调度算法

基于公平原则的调度算法

实时调度

- 非抢占式调度算法

这种调度算法可获得数秒至数十秒的响应时间，可用于要求不严格的实时控制系统中。

- 抢占式调度算法

基于时钟中断

立即抢占

其调度延迟可降为几十至几毫秒，如果是立即抢占则降低至几毫秒至100微秒甚至更低，可用于大多数的实时系统中。

最早截止时间优先EDF算法

任**务时间截止时间越早，其优先级越高**，既**可以用抢占式调度方式，也可以使用非抢占式调度方式**

最低松弛度优先LLF算法

松弛度即任务的紧急度，任务的紧急程度越高，优先级越高（也是可以容缓的时间）

该算法主要用于可抢占调度方式中

松弛程度 = 截止时间 - 运行时间 - 当前时间

死锁的定义

- 如果一组进程中的每一个进程都在等待仅由该组进程中的其它进程才能引发的事件，那么该组进程是死锁的

例如在生产者与消费者问题中：如果生产者在询问缓冲区是否有空位前直接进行互斥访问，那么如果缓冲区没有位置，生产者进程就会堵塞，从而导致跳转到消费者进程，而生产者没有对临界区进行释放，从而导致消费者进程也被阻塞，最终导致，生产者在等待消费者消费产品释放缓冲区，而消费者正在等待生产者释放临界区，最终会导致死锁现象的产生。

产生死锁的必要条件

- 互斥条件，例如生-消问题中，临界区只能由其中一方使用
- 请求和保持条件，在进程占用资源的同时发出对另一个新的资源的请求，将会阻塞请求进程，对自己已获得的资源不放。
- 不可抢占条件，进程获得的资源不能被抢占，只能等自己使用完释放
- 循环等待条件，必然存在一个进程-资源的循环链

处理死锁的方法

- 预防死锁
- 避免死锁
- 检测死锁
- 接触死锁

预防死锁

破坏“请求和保持”条件

资源静态分配法（AND信号量）

- 在所有进程开始之前，一次性地申请在整个运行过程中所需的全部资源
- 但是资源被严重浪费，也会使进程经常发生饥饿现象

改进算法

- 允许进程只获得运行初期所需的资源，便开始运行。这些资源使用完毕并全部释放，再申请新的所需资源。（**借新书之前先还旧书**）

破坏“不可抢占”条件

破坏“循环等待”条件

- 资源顺序分配法，对所有资源进行编号，然后按照升序的顺序请求资源

避免死锁

当系统处于安全状态的时候，**一定不会**发生死锁现象。

当系统处于不安全状态的时候，则**可能发生**死锁现象

安全状态就是系统能按照某种进程推进顺序为每个进程p分配资源，从而使每个进程都可以顺利的完成。

银行家算法

```
//银行家算法中的数据结构
Available[m]//每一个元素代表这一类资源的可用的数目

Max[n][m]//一个nxm的矩阵，代表n个进程中的每一个进程对m类资源的最大需求,例如Max[i][j] = K,则表示i类进程对j类资源的最大需求量为K

Allocation[n][m]//它代表每一类资源已经分配个每一个进程的资源数，例如Allocation[i][j] = K,则表示进程i已经分的j资源的个数为K

Need[n][m]//它代表每一个进程尚且需要各个资源的数目，例如Need[i][j] = K,则表示进程i还需要K个j资源才能完成任务。

//其中:Need = Max - Allocation
```

银行家算法是**试探着**把资源分配给进程，如果系统处于安全状态，则才正式将资源分配给进程P，否则本次**试探**分配将作废。

算法：

（1）如果Request <= Need,跳转到(2)，否则认为出错，因为他所需要的资源已经超过最大值（**因为请求的资源不可能大于它所需要的资源**）

（2）如果Request <= Available,跳转到(3)，否则表示尚无足够的资源，P必须等待

（3）则试探着把资源分配给进程P，并修改下列数据结构：

Available = Available - Request

Allocation = Allocation + Request

Need = Need - Request

（4）最后系统执行安全性算法，判断系统是否处于安全状态，若是安全状态，则正式将资源分配给进程。否则将本次分配作废，恢复原来的资源分配状态。

安全性算法

（1）设置两个向量：

Work：表示系统可提供给进程继续运行所需的各类资源数目，它含有m个元素，在执行算法之前，Work=Available；

Finish：它表示系统是否有足够资源分配给进程，在开始之前做Finish=false；当用足够资源分配给系统时候，在令Finish = true

（2）从进程中找到一个能满足下述条件的进程：

Finish = false；

Need <= Work

若找到，则执行步骤（3），否则则执行步骤（4）

（3）执行：

```
Work[j]=Work[i]+Allocation[i,j];
Finish[i]=true;
go to step 2;
```

（4）如果所有进程的Finish［i］= true都满足， 则表示系统处于安全状态；否则，系统处于不安全状态。

优点：

- 资源利用率比静态资源分配法高，又避免死锁。
  缺点：
- 对资源分配过于保守；计算太多，并且需知道对资源的最大需求量，不太实际。

四、存储器管理

程序的装入

- 绝对装入方式（用于仅能运行单道程序的系统）
- 可重定位装入方式（用于能运行多道程序的系统）
- 动态运行时的装入方式（用于能运行多道程序的系统，**但不允许程序运行时在内存中移动位置**）

动态重定位

地址变换过程是在程序执行期间，随着对每条指令或数据的访问自动进行的，故称为动态重定位。

对换(Swapping)

对换就是把内存中暂时不能运行的进程或者暂时不用的程序和数据，调出到外存上，以便腾出足够的内存空间，再把已具备运行条件的进程或者进程所需要的程序和数据换入内存。**这样做提高了内存利用率，直接提高处理机的利用率和系统的吞吐量。**

对换的类型

- 整体对换，又称为进程对换，其实就是处理机的**中级调度**
- 页面对换，又称部分对换、分段对换

分页与分段存储管理方式

- 分页存储管理方式：将用户程序的地址空间分为若干个固定大小的区域，称为“页”或者“页面”
- 分段存储管理方式：将用户程序的地址空间分为若干个固定大小的段，每段可定义一组相对完整的信息。

分页存储管理的基本方法

页面：分页存储管理将进程的**逻辑地址**分为若干个页，并加以编号。

物理块（块）：把内存的**物理地址**分为若干块，同时编号。

页面大小：一般选择适中，且为2的幂，通常为1 KB ~ 8 KB。

分页的地址结构：前一部分为页号P（12-31位），后一部分为偏移量W（**页内地址**，0-11位）。地址空间最大允许有1M页

地址为A，页面的大小为L，则页号P和页内地址d可按下式求得：

$$P = INT\left[\frac{A}{L}\right]$$

$$d = [A]MODL$$

其中：INT是整除函数.

例如系统的页面大小为1KB，设A=2170B，求P和d

首先d = [A] mod L = 2170B mod 1KB = 2170B mod 1024B

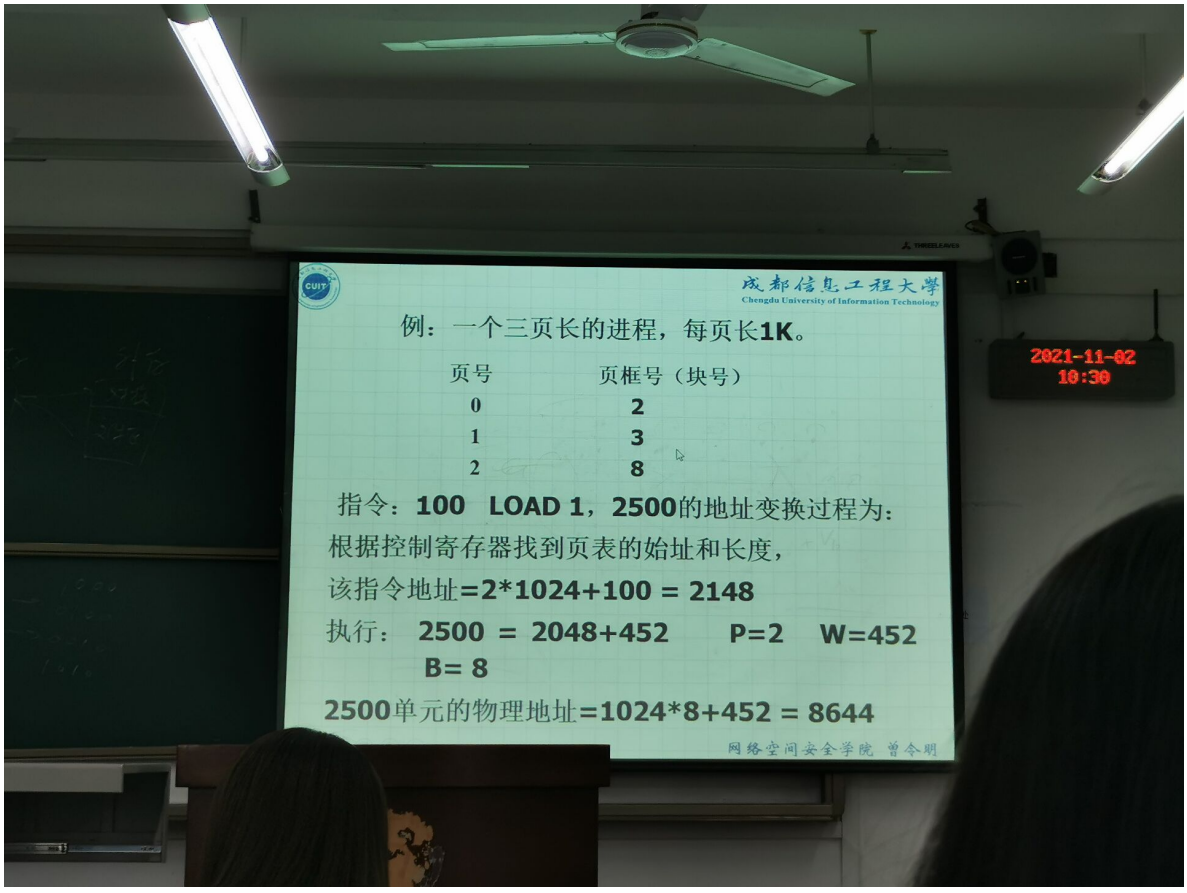
= 2170-2048 = 122

P = INT[A/L]=INT[2170B / 1024B] = 2

地址变化机构

当进程要访问某个逻辑地址中的数据时，分页地址变换机构会自动地将有效地址(有效地址)分为页号和页内地址两部分，如果页号小于页表长度，那么将页表初始地址与页号和页表项长度的乘积相加，便得到该表项在页表中的位置。

例如下图：



五、虚拟存储器

虚拟存储器的定义

- 是指具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的一种存储器系统。

虚拟存储器的特征

- 多次性
- 对换性
- 虚拟性：虚拟性以多次性和对换性为基础（**是虚拟存储器最本质、最重要的特征**）
- 离散性：多次性和对换性必须建立在离散性的基础之上

程序运行时的局限性

- 时间局限性
    - 如果程序中的某条指令被执行，那么不久之后这条指令可能就会被再次执行，同理如果某个数据被访问过，那么之后这个数据可能被再次访问，这就是时间局限性，**典型原因的是程序中存在着大量的循环操作**
- 空间局限性
    - 一旦程序访问了某个存储单元，那么不久之后，其附近的存储单元也可能被访问，即程序在一段时间内所访问的地址可能集中在一定的范围之内，**典型原因是因为程序的顺序执行**

请求分页存储管理方式

请求分页系统中的每个页表应含以下诸项

页号	物理块号	状态位**P**	访问字段**A**	修改位**M**	外存地址

- 状态位P，指示该页是否调入内存，供程序访问时参考。
- 访问字段A，用于记录本页在一段时间内被访问的次数，或者最近未被访问的时间，供页面置换算法选择换出页面时参考。
- 修改位M，表示该页在调入内存后是否被修改过
- 外存地址，指出该页在外存上的地址，通常是物理块号，供调入该页时使用。

缺页中断机构

当所要访问的页面不在内存时，便产生一缺页中断，请求OS将所缺之页调入内存。它与一般的中断有着明显的区别，主要表现在下面两个方面

- 在指令执行期间产生和处理中断信号
- 一条指令在执行期间可能产生多次缺页中断

页面置换算法

- 最佳置换算法（Optimal）
    - 原理：调入一页而必须淘汰一个旧页时，所淘汰的页应该是以后不再访问的页或距现在最长时间后再

访问的页。
    - 通常可以获得最低的缺页率，但算法无法实现（无法预知在进程中那个页面是未来最久时间不再被访问）。
    - **在页面序列中找最右边的淘汰**
- 先进先出页面置换算法（First In First Out，FIFO）
    - 原理：总是淘汰最先进入内存的页面，即选择在内存中驻留时间最久的页面予以淘汰。
    - 最早出现的置换算法，算法实现简单，但不能保证部分页面（含有全局变量、常用函数、例程的页面）不被淘汰。
    - **在页面序列中找连续块数最多的那个淘汰**
- 最近最久未使用置换算法（Least Recently Used，LRU）
    - 原理：根据页面调入内存后的使用情况做出决策，选择最近最久未使用的页面予以淘汰。
    - **找想要添加到队列的页面，离得最远的那个淘汰**
- LRU置换算法的硬件支持
    - 寄存器
    - **找到具有最小数值的寄存器所对应的页面进行淘汰(即二进制最小的)**
- 最少使用置换算法（Least Frequently Used，LFU）
    - 采用移位寄存器方式，选择在最近时期使用最少的页面作为淘汰页。

利用“L=S”准则调节缺页率

- L为缺页之间的平均时间，S为平均缺页服务时间，即用于置换一个页面所需的时间，如果 L >> S ,那么很少会发生缺页，但是此时磁盘未得到充分利用，if L < S，则频繁发生缺页，缺页速度已经超过磁盘处理能力。所以只有L接近S时，磁盘和处理机都能得到充分利用。

六、输入输出系统

块设备与字符设备

- 块设备：是指数据的存取和传输都是以数据块为单位的设备。**典型的块设备是磁盘**
- 字符设备：是指数据的存取和传输是以字符为单位的设备,如**键盘、打印机等**

DMA方式（直接存储器访问方式）

- I/O设备能直接与主存交换数据而不占用CPU
- 特点
  - 提高了CPU的利用率数据
  - 传输的基本单位是数据块：在主机与I/O设备之间每次至少传递一个数据块
  - 所传送的数据块是从设备直接送入内存
  - 传送过程是在控制器的控制下完成，在传送一个或多个数据块的开始和结束才需CPU干预

DMA控制器

- 命令/状态寄存器CR，用于接收从CPU发来的I/O命令，或有关控制信息，或设备的状态
- 内存地址寄存器MAR，存放数据从设备传送到内存的起始目标地址，输出时，它存放由内存到设备的内存源地址。
- 数据寄存器DR，用于暂存从设备到内存，或从内存到设备的数据。
- 数据计数器DC，存放本次CPU要读或写的字节数

DMA工作流程

假脱机（SPOOLing系统）

特点

- 提高了I/O速度
- 将独占设备改造为共享设备
- 实现了虚拟设备功能

假脱机打印机系统的工作流程

- 假脱机管理进程在磁盘缓冲区中为之申请一个空闲盘块，并将要打印的数据送入其中暂存
- 管理进程然后再为用户进程申请一张空白的用户请求打印表，并将用户的打印要求填入其中，将此表挂到

假脱机文件队列上
- 然后当打印机空闲的时候，假脱机的打印进程，从假脱机文件队列的队首摘取一张请求打印表，然后将想要打印的数据由输出井传送到内存缓冲区，在交付打印机进行打印。

磁盘调度

- 先来先服务(First-Come First-Served，FCFS)
  - 原理：根据进程请求访问磁盘的先后次序进行调度。
  - 最简单的调度算法，磁盘移臂调度采用的是先来先服务算法。
  - 优点：公平、简单、不会出现饥饿现象
  - 缺点：平均寻道时间可能较长
- 最短寻道时间优先(Shortest Seek Time First，SSTF)
  - 原理：选择有距当前磁头所在磁道最近的访问磁道的进程。
  - 优点：比FCFS有更好的寻道性能
  - 缺点：可能导致某个进程发生饥饿现象
- 扫描(SCAN)算法，又称电梯调度算法
  - 原理：选择与当前磁头移动方向一致且距离最近的进程。
  - 优点：寻道性能较好，避免出现“饥饿”现象
- 循环扫描(CSCAN)算法
  - 原理：规定磁头单向移动。

七、文件管理

文件和文件系统

- 记录
  - 记录是一组相关数据项的集合，用于描述一个对象在某方面的属性。
- 文件
  - 由创建者所定义的、具有文件名的一组相关元素的集合。可分为有结构文件和无结构文件两种。
- 关系

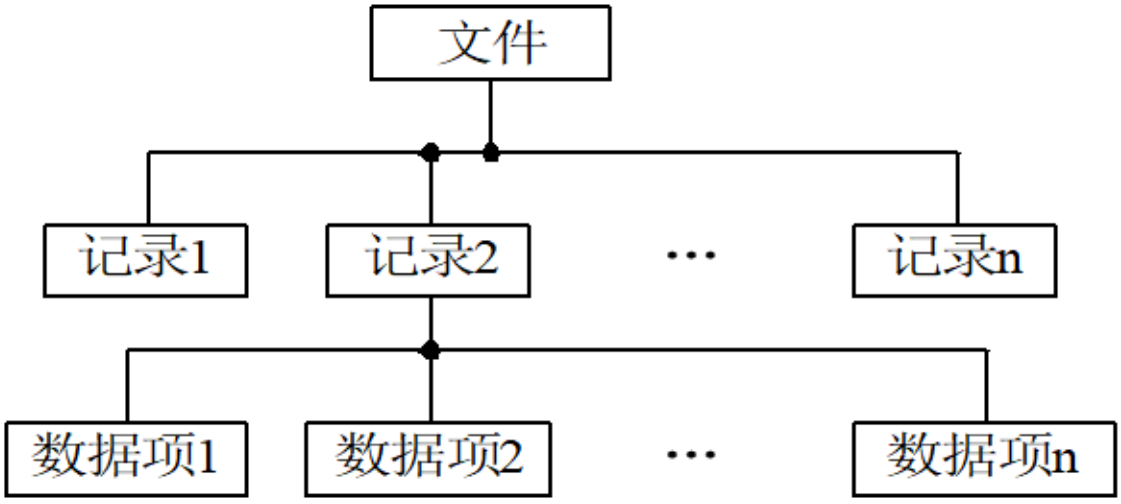


图 7-1 文件、 记录和数据项之间的层次关系

文件的逻辑结构

- 是从用户观点出发所观察到的文件组织形式，即文件是由一系列的逻辑记录组成的，是用户可以直接处理

文件逻辑结构的分类

- 按是否有结构分类
  - 有结构文件：由一个以上的记录构成的文件，如定长记录，变长记录。
  - 无结构文件：由字符流构成的文件，如流式文件。
- 按文件的组织方式分类
  - 顺序文件：记录按某种顺序排列所形成的文件。
  - 索引文件：可变长记录文件建立一张索引表，为每个记录设置一个表项，以加速对记录的检索速度。
  - 索引顺序文件：顺序文件和索引文件相结合，为一组记录中的第一个记录建立索引表项。

文件的物理结构

- 是文件存储在外存上所形成的一种存储组织形式，是用户看不见的，且与存储介质的存储性能有关。

索引顺序文件的特征：

- 记录方式是按照关键字的顺序组织起来的
- 通过文件索引表，可以实现对索引顺序文件的随机访问
- 通过溢出文件，用来记录新增加、删除和修改的记录

目录管理的主要要求：

- 实现“按名存取”

- 提高对目录的检索速度
- 文件共享
- 允许文件重名

八、磁盘存储器的管理

FAT技术

原理

- 在每个分区中配有两张相同的文件分配表FAT1和FAT2。在FAT的每个表项中存放下一个盘块号，实际上是用于盘块之间的链接的指针。通过这个指针可以将一个文件的所有盘块链接起来。